Arpan Mangal
Deepanshu Jindal

Assignment 3

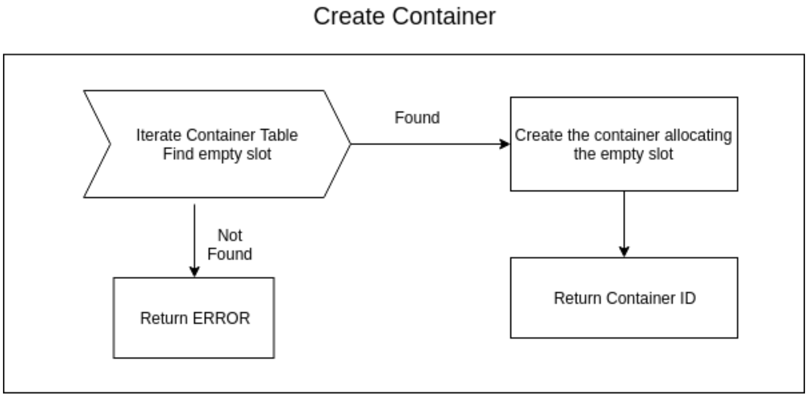
COL331
April 25, 2019

1 Container Manager

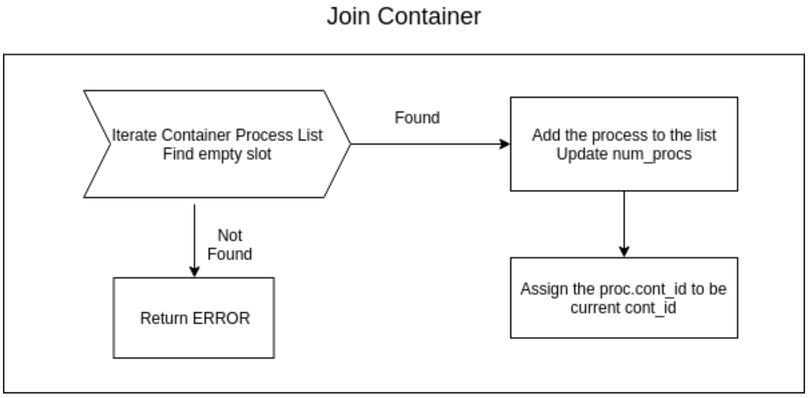
- This is implemented in the Kernel space.
- A container table is stores all containers and relevant data structures for them. This table is analogous to the process table. The container data structure:

```
struct container_desc{
  int allocated;        // whether the container is allocated
  int procs[MAX_PROCS]; // List of processes in this container
  int num_procs;        // Number of processes in this container
  int last_run_id;      // PID of last scheduled process of the container
};
```

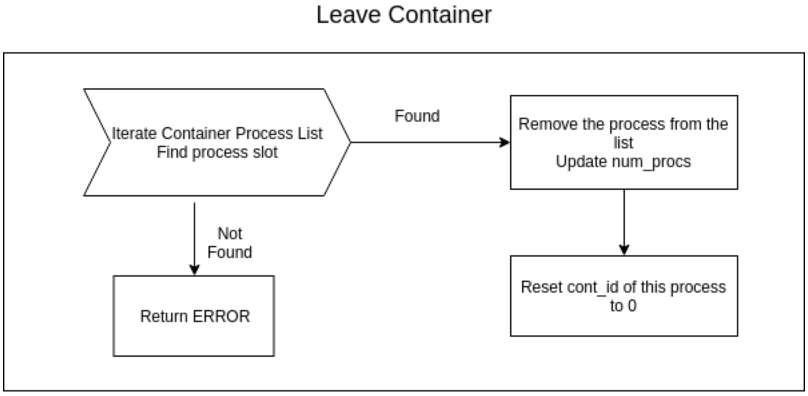
1.1 Container Creation



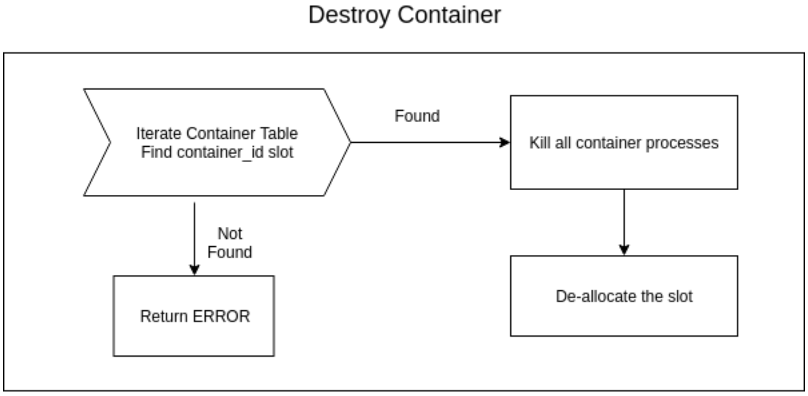
1.2 Joining a Container


Join Container
Iterate Container Process List
Find empty slot
Found
Add the process to the list
Update num_procs
Not Found
Return ERROR
Assign the proc.cont_id to be current cont_id

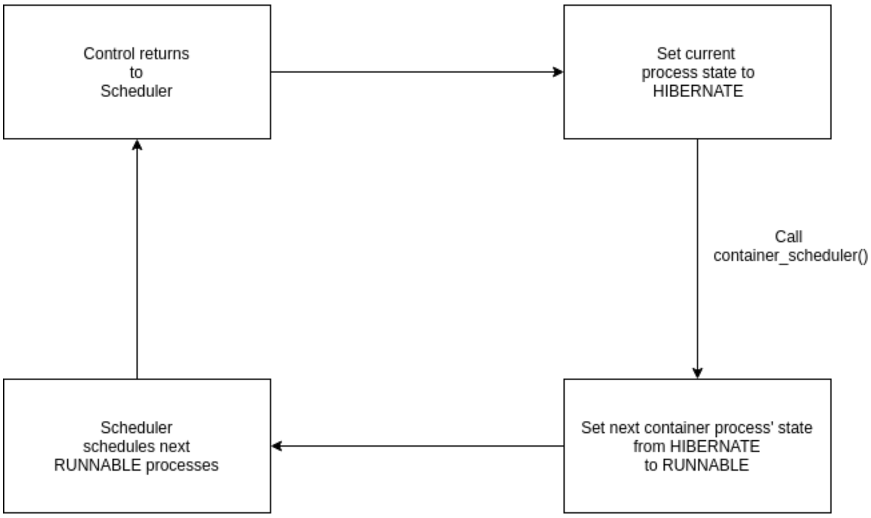
1.3 Leaving a Container


Leave Container
Iterate Container Process List
Find process slot
Found
Remove the process from the list
Update num_procs
Not Found
Return ERROR
Reset cont_id of this process to 0

1.4 Destroying Container



2 Virtual Scheduler


Control returns to Scheduler
Set current process state to HIBERNATE
Call container_scheduler()
Set next container process' state from HIBERNATE to RUNNABLE
Scheduler schedules next RUNNABLE processes

We have implemented a intra-container round robin scheduling algorithm. Each container iterates over its contained processes in a round robin fashion and makes the next ready process available for scheduling.

1. The currently scheduled process returns the control back to scheduler.
2. If the returned process state is RUNNABLE, scheduler sets it to HIBERNATE, and *container_schedule()* is called with the current container id.
3. *container_schedule()* sets the state of next container process which was on HIBERNATE to RUNNABLE.
4. Control returns to scheduler which schedules the next RUNNABLE process.

2.1 ps syscall

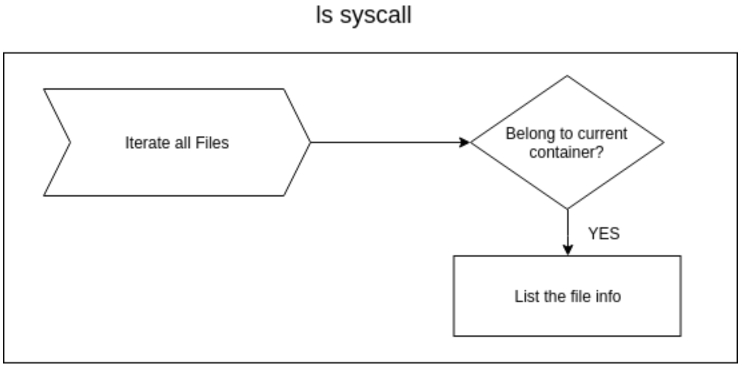
To implement ps syscall respecting container behaviour, we modified the standard ps syscall implemented in assignment 1 to check for container id of all the active processes and then print only those processes whose container id was same as current process.

3 Virtual File system

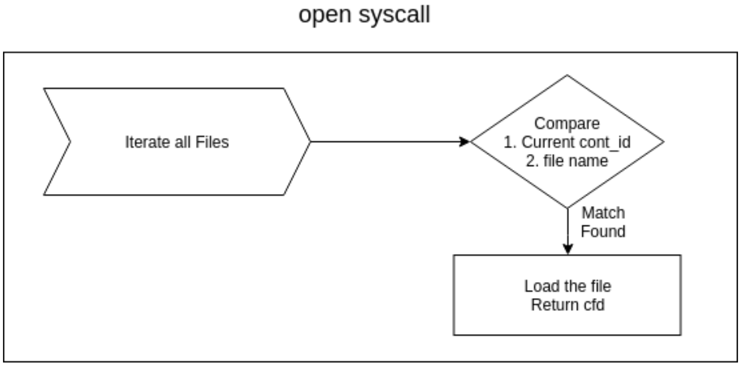
We have an implicit table which produces a mapping from a given file name and the container ID of the process to give a unique **cfd**.

cfd	File Name	Container ID
2	my_file	1
5	my_file	2
8	file_3	1

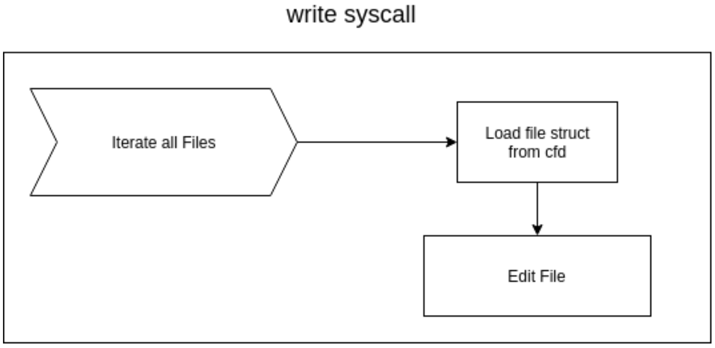
3.1 ls syscall



3.2 open syscall



3.3 write syscall


write syscall
Iterate all Files
Load file struct from cfd
Edit File